

REMARKS

In the Office Action dated September 11, 2002, the drawings and specification were objected to; claims 1-20, 22-24, 27, and 28 were rejected under 35 U.S.C. § 103 over U.S. Patent NO. 5,857,192 (Fitting) in view of U.S. Patent Application Publication No. 2002/0023175 (Karlak); and claims 21, 25, and 26 were rejected under § 103 over Fitting in view of Karlak in further view of U.S. Patent No. 5,848,410 (Walls).

OBJECTIONS TO SPECIFICATION AND DRAWINGS

The descriptions on pages 3, 5, and 6 have been amended to address the objections.

REJECTIONS UNDER 35 U.S.C. § 103

As amended, claim 1 recites a method of performing a test that comprises: in each of first and second test systems, identifying a file name of a first data file to use in each of first and second tests based on plural received parameters; and in each of the first and second test systems, using the first data file in performing the respective one of the first and second tests. As recited in claim 1, the first and second test systems effectively share the same data file, namely the first data file.

In contrast, in Fitting, an empty file is generated by a test system to send to a database server. The database server, from this empty file, generates a database query to obtain a model number. This model number is provided back to the requesting test system by appending the model number to the file name of the empty file. In this way, the products that are being tested do not need to store their model information, since the test systems are able to retrieve the model information based on the identifier of each product.

There is nothing in Fitting to even remotely suggest that first and second test systems use the same data file in performing respective first and second tests. There is also nothing in Karlak that suggests such a modification, or that suggests the combination of Fitting with Karlak. Karlak describes a system that operates multiple applications that reside on one or more computer systems and that may be operated by placing objects into a queue and allowing application interfaces that run the applications to retrieve objects from the queue when the application is available for operation. *See* Karlak, Abstract. The multiple application controller

200 of Karlak is able to run one or more strategies defined against input records or files provided by a user. See Karlak, Paragraphs 0036-0046. However, Karlak has nothing to do with identifying a file name of a first data file to use in first and second tests by first and second test systems based on received parameters, or using the first data file identified based on the received plural parameters in performing the first and second tests by the first and second test systems.

Therefore, even if they are combined, there is no teaching or suggestion of the combination of elements recited in claim 1 in Fitting and Karlak. Therefore, for at least this reason, a *prima facie* obviousness rejection does not exist with respect to claim 1. Also, there is simply no motivation to combine Fitting and Karlak. Karlak relates to selecting a strategy based on user inputs to run multiple applications. No mention is made whatsoever of performing tests using a data file identified based on received plural parameters. It is unclear what motivation or suggestion exists, whether implicit or explicit, in either Fitting or Karlak, for combining their teachings. The two references are directed to completely different applications; therefore, a person of ordinary skill in the art would not have been motivated to combine the teachings of Fitting and Karlak to achieve the claimed combination.

Independent claim 6 is also allowable over the alleged the combination of Fitting and Karlak. As conceded by the Office Action, Fitting teaches that the second parameter indicates a file type, not the name of the database on which to perform a test. Instead, reliance was made on Karlak for this teaching. Even if Karlak can be properly combined with Fitting, there is simply no teaching of *combining first and second values* to generate a file name of a test file to use in a test in Karlak. Therefore, a recited element of claim 6 is missing from the alleged combination of Fitting and Karlak. Also, as noted above, there is no motivation of suggestion to combine Karlak and Fitting.

With respect to independent claim 14, there is no teaching or suggestion in the alleged combination of Fitting and Karlak of a routine executable on a control unit to receive a first parameter and a second parameter, to *combine* the first and second parameters to form a string, and to *identify a file name of the data file using this string*, and a *test module executable on the control unit to perform the test using the data file*.

In Fitting, the empty file that is generated by the test system in response to testing a product contains a name that has a product identifier. This empty file is provided to the database

server, which generates a query to retrieve from a database the model number corresponding to the product identifier. At that time, the name of the empty file is changed to add the model number to the end of the product identifier. In this way, the test system of Fitting is able to determine the model number of a given product. There is nothing to suggest that a test module performs a test using the data file that is identified based on a string from the combination of first and second parameters. In fact, the opposite is suggested, as the empty file is designed to contain no information. There is also no teaching of the missing element in Karlak. Therefore, the alleged combination of Karlak and Fitting does not teach or suggest claim 14. Also, there is no motivation to combine Karlak and Fitting.

Independent claim 22 recites identifying one of plural data files having a name containing a string (that is combined from first and second parameters) for use in a test procedure. This act is not taught or suggested by either Fitting or Karlak, either alone or in combination.

Independent claim 23 recites a method of performing a test that comprises concatenating first and second parameters to generate a string that is at least a portion of a file name, and searching a predetermined directory on a device to find a test file containing the string. This is not taught or suggested by the alleged combination of Fitting and Karlak. Independent claims 27 and 28 are also similarly allowable over the alleged combination of Fitting and Karlak.

Allowance of all claims is respectfully requested. The Commissioner is authorized to charge any additional fees and/or credit any overpayment to Deposit Account 20-1504 (MCT.0134US).


Respectfully submitted,

Date: Dec. 11, 2002

PATENT TRADEMARK OFFICE



21906



Dan C. Hu
Registration No. 40,025
TROP, PRUNER & HU, P.C.
8554 Katy Freeway, Suite 100
Houston, Texas 77024
(713) 468-8880 [Phone]
(713) 468-8883 [Fax]



APPENDIX

IN THE SPECIFICATION:

Amend the paragraph starting on page 3, at line 29:

--The test system 12 or 14 also includes a network interface 112 coupled to the network 16. One or more protocol layers 114 are provided above the network interface 112. For example, the protocol layers [14]114 may include an Ethernet layer and an Internet Protocol (IP) layer.--

Amend the paragraph starting on page 5, at line 26:

--If the database name has been received (at 312 or 316), then the value of DBNum is set (at 318) to the appropriate value (1 for TEST or 2 for DEVL). Next, a parameter FString is set (at 320) to the concatenation of the Clarify and DBase parameters, that is,

FString = Clarify, DBase.--

Amend the paragraph starting on page 6, at line 1:

--The data source routine 124 then performs (at 322) a directory list command on the server 110 in which the common set of data files 24 are kept. The output of the directory command is routed to a file FILE.TXT. The file FILE.TXT is then opened and a search is performed (at 324) to find file names that contain the string FString.--

VERSION OF CLAIMS INDICATING CHANGES

Claims 15 and 16 have been cancelled. New claims 29-32 have added. Amend the following claims as indicated (unamended claims in smaller font):

- 1 1. (Amended) A method of performing a test, comprising:
- 2 performing a first test with a first test system;
- 3 performing a second test with a second test system:
- 4 in each of the first and second test systems, receiving plural parameters;
- 5 [and]

6 in each of the first and second test systems, identifying a file name of a
7 first data file to use in each of the first and second tests based on the plural parameter;
8 and
9 in each of the first and second test systems, using the first data file in
10 performing the respective one of the first and second tests.

1 2. The method of claim 1, further comprising performing at least another test with at least
2 another test system using the data file.

1 3. The method of claim 1, further comprising, in each of the first and second test systems,
2 accessing a storage system over a network to find a file name containing strings in each of the plural
3 parameters.

1 4. The method of claim 3, wherein accessing the storage system comprises accessing the
2 storage system to find a file name containing a concatenation of the strings.

1 5. The method of claim 1, wherein each of the tests is performed on a database, and wherein
2 one of the parameters represents the database.

1 6. A method of performing a test, comprising:
2 receiving a first value;
3 receiving a second value representing a database to perform a test on; and
4 combining the first value and the second value to generate a file name of a test file to use
5 in the test.

1 7. The method of claim 6, wherein receiving the test value comprises receiving a
2 predetermined string, the predetermined string being part of the file name of the test file.

1 8. The method of claim 6, further comprising performing the test using a test module and
2 invoking a routine, from the test module, to generate the file name of the test file.

1 9. The method of claim 8, further comprising executing the test module in a test system.

1 10. The method of claim 9, further comprising the test module performing a test on the
2 database coupled over a network.

1 11. The method of claim 6, further comprising performing the test using a first test system,
2 wherein the receiving and combining acts are performed in the first test system.

1 12. The method of claim 11, further comprising, in a second system:
2 receiving the first value;
3 receiving the second value representing the database;
4 combining the first value and the second value to generate the file name of the test file;
5 and
6 performing another test on the database using the test file.

1 13. The method of claim 12, wherein the first test system performs a first type of test and the
2 second test system performs a second type of test.

1 14. (Amended) A test system comprising:
2 an interface to a network coupled to a storage unit containing a data file
3 for use in a test;
4 a control unit;
5 a routine executable on the control unit to receive a first parameter and a
6 second parameter and to combine the first and second parameters to form a string,
7 the routine to identify a file name of the data file based on the string; and
8 a test module executable on the control unit to perform the test using the
9 data file.

1 17. The test system of claim 14, wherein the routine is executable to access the storage unit
2 and to search file names on the storage unit for a file name containing the string.

1 18. (Amended) The test system of claim 14, [further comprising a]wherein the
2 test module is executable on the control unit to perform a test of a database coupled to the
3 network, the second parameter representing the database.

1 19. The test system of claim 18, wherein the test module is executable to pass the first and
2 second parameters to the routine.

1 20. The test system of claim 19, wherein the routine is executable to prompt a user for one or
2 both of the first and second parameters if not passed by the test module.

1 21. The test system of claim 20, wherein the routine is executable to set a file name of a
2 default data file if not received from the test module or the user.

1 22. (Amended) An article comprising at least one storage medium containing
2 instructions that when executed cause a system to:
3 combine a first parameter and a second parameter to form a string;
4 access a storage unit over a network, the storage unit containing plural
5 data files; and
6 identify one of the data files [based on]having a name containing the string
7 [to] for using in a test procedure.

1 23. A method of performing a test, comprising:
2 receiving a first parameter containing a predetermined value;
3 receiving a second parameter representing a database to perform a test on;
4 concatenating the first parameter and the second parameter to generate a string that is at
5 least a portion of a file name; and
6 searching a predetermined directory on a device to find a test file containing the string.

1 24. The method of claim 23, further comprising accessing the device over a network to
2 search the predetermined directory.

1 25. The method of claim 23, further comprising:
2 prompting a user for a value of the first parameter; and
3 setting a default value for the first parameter if the first parameter value is not received
4 from the user.

1 26. The method of claim 25, further comprising:
2 prompting the user for a value of the second parameter; and
3 setting a default value for the second parameter if the second parameter value is not
4 received from the user.

1 27. A system comprising:
2 an interface to a network coupled to a storage unit containing a directory of data files;
3 a control unit;
4 a routine executable on the control unit to receive a first parameter and a second parameter and to
5 concatenate the first and second parameters to form a string, the first parameter containing a predetermined
6 value, and the second parameter representing a database to perform a test on,
7 the routine executable to search the directory to find a file name of one of the data files
8 that contains the string and to set the one data file as the data file to use for the test; and
9 a test module executable on the control unit to perform the test.

1 28. A method of performing tests, comprising:
2 receiving a predetermined common parameter;
3 receiving a second parameter representing a database to perform a test on;
4 concatenating the common parameter and the second parameter to generate a string that
5 is at least a portion of a file name; and
6 searching a predetermined directory on a device to find a test file containing the string,
7 wherein receiving the common parameter, receiving the second parameter, concatenating
8 the common parameter and the second parameter, and searching the predetermined directory is performed
9 in each of plural test systems.

1 29. (New) The method of claim 1, further comprising:
2 combining the plural parameters to form a string; and
3 locating the first data file by finding the file name containing the string.

1 30. (New) The method of claim 6, further comprising locating the test file
2 having the file name.

1 31. (New) The test system of claim 14, the routine to locate the data file by
2 finding the file name containing the string.

1 32. (New) The method of claim 23, wherein searching the predetermined
2 directory comprises searching the predetermined directory to find the test file having a
3 name containing the string.